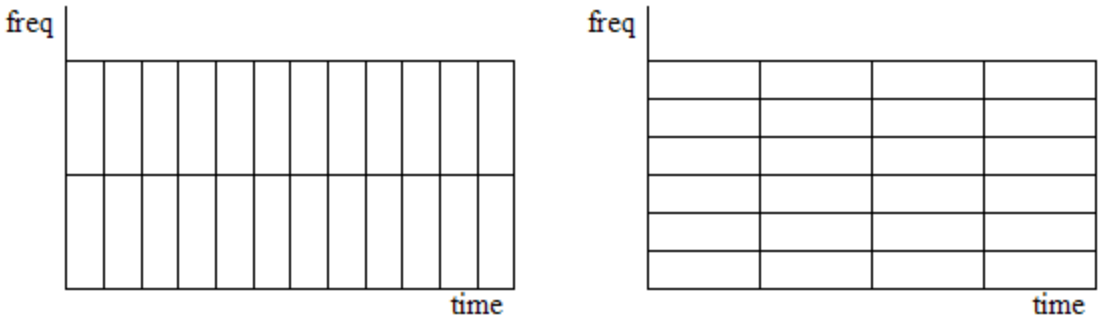
Introduction
Motivation for Music Transcription through Frequency Reassignment

Imagine yourself as a musician, and you hear a song that you would like to play/practice. Either you can go to a music store and buy the score, or order it online; HOWEVER, there is a third option: you can use a program that will take an audio file as an input, and output the notes of the piece. Our group project forms a foundation for such a musical transcription program.

Background

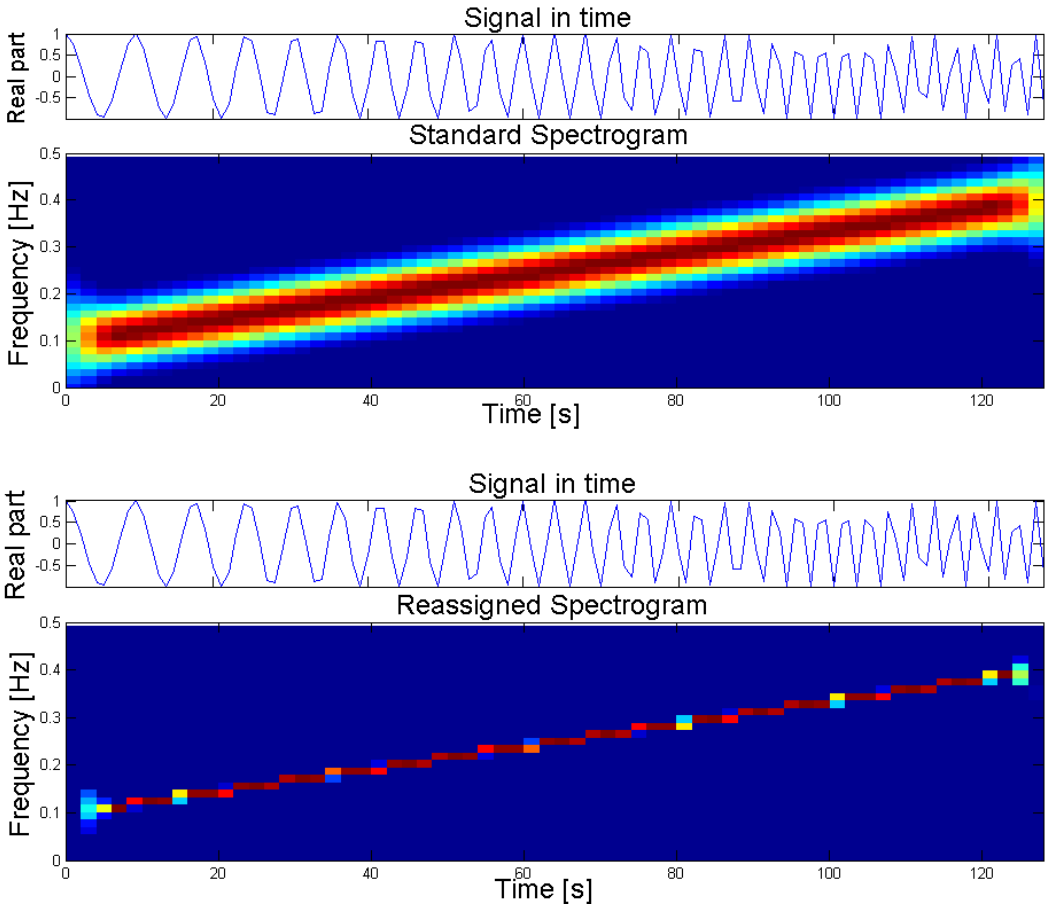
Background on analyzing signals in the time-frequency representation.

The most natural way of analyzing the notes of an audio signal is through a time-frequency representation of it. This time-frequency representation is known as a spectrogram. However, this spectrogram has some resolution issues stemming from the windowing operation used on the signal in order to compute the STFT (Short-Time Fourier Transform) of it, putting it in the form we see: time vs. frequency. This windowing has an effect not unlike the Heisenberg Uncertainty Principle. A wide window improves the frequency resolution, and a narrow window improves the time resolution, thus we can never achieve a higher resolution in both the frequency and time domain (shown in figure below). However, we need to achieve a high resolution in both to analyze the notes with great precision.



Frequency Reassignment

There are a few different methods of achieving this high resolution of the spectrogram; but, one of the most successful in terms of implementation (and the one we used in our project) is known as the Reassignment Method. Worked on by Patrick Flandrin, Francois Auger, and other various people around the world, the Reassignment Method is quite simple in theory, but the implementation and the math behind it are indeed quite complicated. The Reassignment Method takes a “blurry” spectrogram, finds the spot with the most magnitude at each time instant, and reassigns the frequency to this single value.



Above is the Reassignment method implemented and computed for the very well-known chirp signal. The new, reassigned spectrogram is a much higher resolution than previous one. The wide line of a signal compresses down into a very thin line. Finding the “center of gravity” of the frequency greatly increases the resolution and thus, we are able to analyze notes with much

greater precision! With frequency reassignment in our toolbox, we can now continue along the road to reach our overall goal of musical transcription.

Preparing the Signal
This module discusses the process of preparing an audio signal for music transcription.

The Method of Music Transcription: Preparing the Signal

Each “module” discussed on this page and the next is a different function in MATLAB.

Calculation of Reassigned Time-Frequency Representation

The method of music transcription is divided into different modules, called by a top module when necessary. First of all, we will need to calculate the reassigned spectrogram using a separate module. The output of the frequency reassignment module is a matrix with n columns and m rows. The columns represent the different time instants and the rows represent the different frequencies, both positive and negative. Each element in the matrix corresponds to the magnitude at that specific frequency at that specific time instant. The code for calculating reassigned spectrograms comes from the Time-Frequency Toolbox for MATLAB created by Patrick Flandrin and Francois Auger.

Eliminating Negative Frequencies

On the top module, all other modules are called and several other functions are made. First we will call the frequency reassignment function to calculate the reassigned matrix of our input signal (the song). Once we have obtained the matrix, we need to get rid of all negative frequencies (since we only need to analyze the positive ones to determine the note). We do this by using a Hilbert transform, which is basically a filter that only passes positive frequencies. The matrix is now ready to be analyzed.

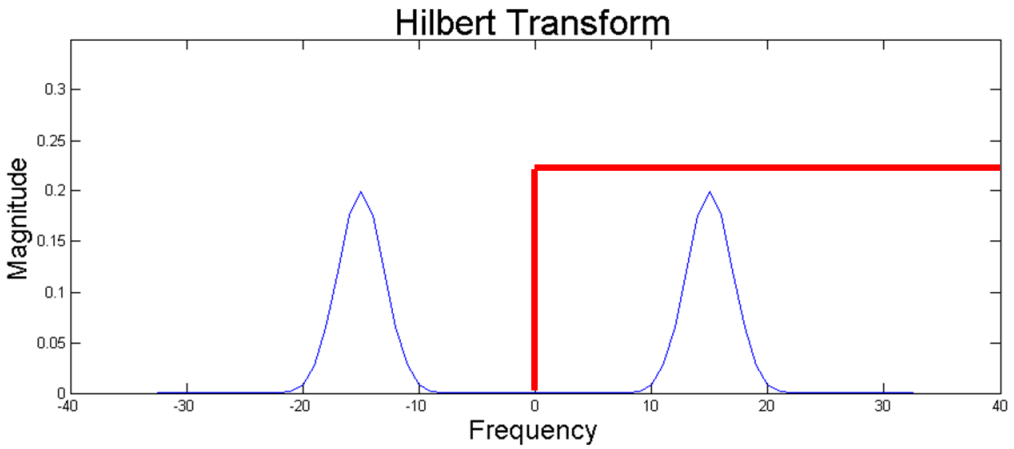


Fig. 1 A Hilbert Transform eliminates the negative frequency component

Analyzing the Signal
This module discuss the process of analyzing an audio signal for music transcription.

The Method of Music Transcription: Analyzing the Signal

Distinguishing Between Notes

Our second module (get_note) is the one that will analyze the signal at a specific time, split the signal into the different notes, and get the frequencies of each note. Since each element of the matrix corresponds to a magnitude, the frequency with the maximum magnitude at a specific time instant represents the frequency of the signal at that time. Therefore, we need to check the maximum magnitude for every column and determine the frequency. Sometimes, however, some columns will not have a frequency because no note is being played ag that time instant, but it will still have a maximum magnitude at a frequency because it has noise. To deal with this, we have set a threshold magnitude that determines whether a column has a note to be analyzed or not. If the maximum magnitude at a column is below the threshold, the frequency at that time will not be considered a note, therefore it will not be analyzed.

To identify a note, we use simple edge detectors. When the signal changes frequency, the program will analyze each column until it detects another change in frequency. The range of time instants obtained from this process will represent a single note, and the frequency of the note will be the average of the maximum amplitude frequencies of each time instant in the given range. Each frequency of each note will be inputted to the identify_note module one at a time for it to output the note name. To prevent any imperfections, we have also created a time threshold, such that, if the length of a note (time instants between edge detections) is lower, such note will not be analyzed because it will not be considered a note.

NOTE: The frequency and duration thresholds work together to distinguish between notes and noise. There may be a few consecutive time instants for which noise is greater that the amplitude threshold. However, no signal should have noise that is consistently greater than the amplitude threshold

for the duration of the duration threshold or longer. As such the two thresholds work in tandem to eliminate notes from the transcription process.

NOTE: The duration threshold does involve a tradeoff: while noise is eliminated, the user is prevented from analyzing a file that contains notes that are less than 100 time instants long. Any such note would be treated as noise. As such, the program may be unable to correctly handle very fast rhythms in low quality signals.

Identifying Notes

Our next module will basically identify the note. It takes a frequency as an input and it outputs a string with the name of the note being played. We have created a library with different frequency ranges that correspond to different notes. The module will find the range to which the input frequency corresponds and input output the string for that range. This process will be done for every note that is given by the get_note module.

This identification process would not be possible if we only had a regular spectrogram, since the precision is very low, causing a very high error that would lead to wrong identification of a note. The frequency reassignment process is an extremely essential part of this method.

Generating Synthetic Audio Files

As an extra module, we have created a simple function to create tones, in order to test our project. In this function, we have different square waves with different frequencies. We simply create a vector with the notes that we want and we have a song to be analyzed.

Simple Input
The results of the Project Treble Maker system for a simple input.

Base Input

We developed the music transcription system by analyzing a simple input of a “C5” followed by an “E5.” The letter corresponds to the musical note, and the number corresponds to the octave (a “C4” is lower than a “C5” and so forth; a “C4” is “Middle C”). The input is comprised purely of tonal components, and no chords are used

Traditional Spectrogram

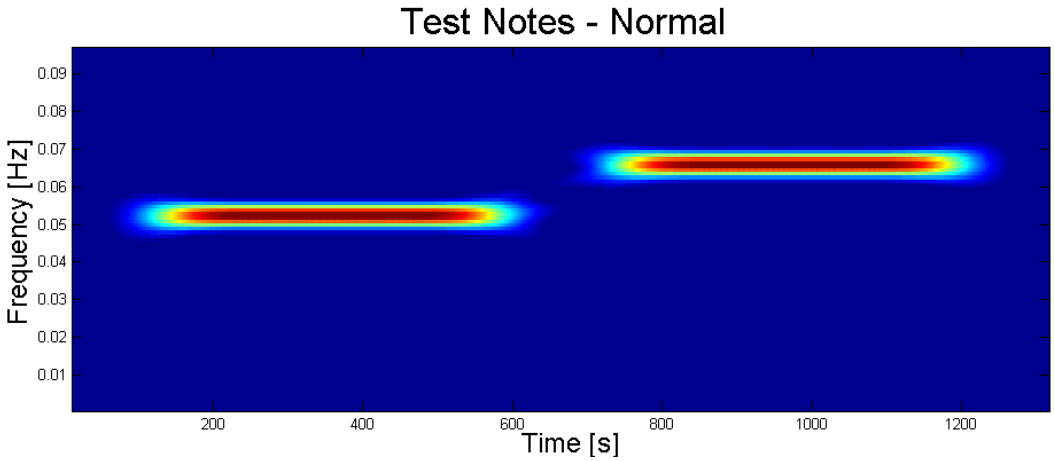


Fig 1 The traditional “smeared” spectrogram of the base input

Reassigned Spectrogram

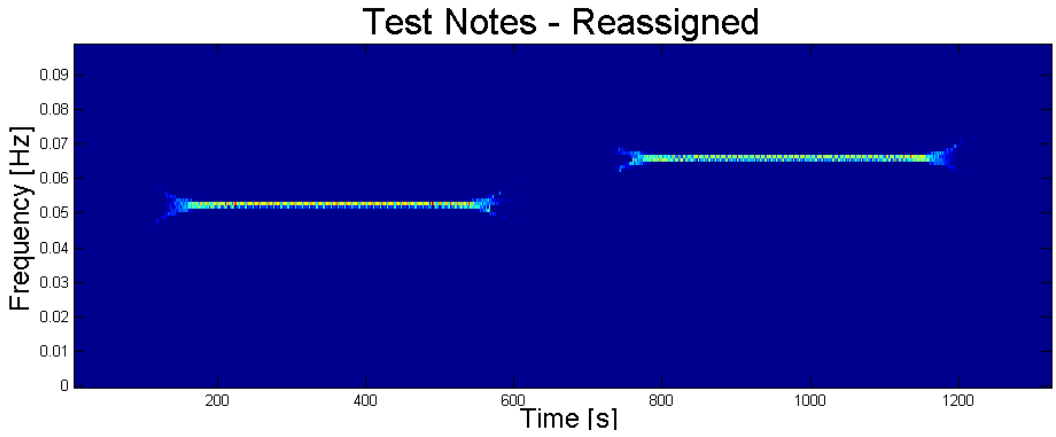


Fig 2 The higher-resolution "Reassigned" spectrogram of the same input

System Output

Ultimately, our system correctly identified the two notes being played as a "C5" followed by and "E5." The output format from MATLAB is shown below:

```
EDU>> treblemaker(song,1024)

notes =

    'C5 E5 '
```

Fig 3 MATLAB prints out the correct result

Complex Input
The module contains the results of the Project Treble Maker system being run on a complex input.

Complex Input

Once the program was successfully developed, it was run on a recognizable tune: the first four measures of “Twinkle Twinkle Little Star.” Again, the traditional spectrogram, the reassigned spectrogram, and the MATLAB output are shown below. Each note of the song was correctly identified and output by the system.

Traditional Spectrogram

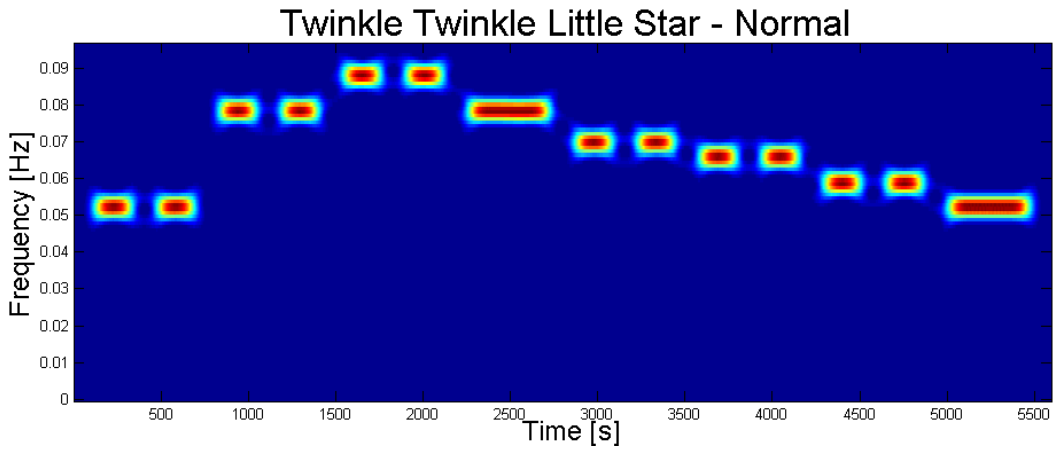


Fig 1 The traditional “smeared” spectrogram of “Twinkle Twinkle Little Star”

Reassigned Spectrogram

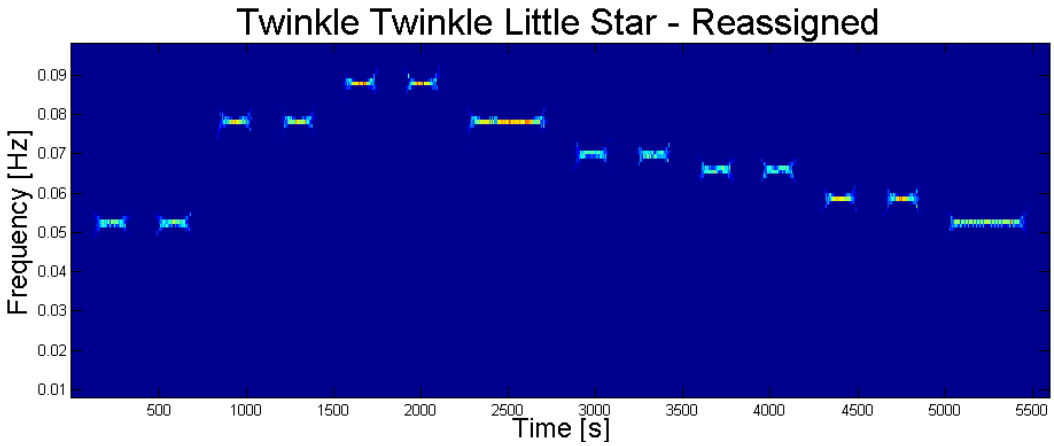


Fig 1 The reassigned spectrogram of “Twinkle Twinkle Little Star”

System Output

Again, the system correctly identified all of the notes of the input signal without error. The first four measures of “Twinkle Twinkle Little Star” are as follows:

Here is the MATLAB output:

```
EDU>> treblemaker(song,1024)

notes =

    'C5 C5 G5 G5 A5 A5 G5 F5 F5 E5 E5 D5 D5 C5 '
```

Fig 3 The MATLAB output accurately identifying each note

Results
This module summarizes the results of Project Treble Maker.

Results

Ultimately, we succeeded in creating a foundational music transcription system. The system successfully evaluates signals of a certain type of input and outputs the notes that correspond to that signal. The music transcription system we developed is currently limited to inputs with the following characteristics:

- Tonal inputs (as opposed to a hand clap, a drum hit, etc.)
- Single notes (as opposed to chords)
- Notes of at least 100 time instants in duration

Other limitations include:

- Size of the MATLAB signal vector (tradeoff between signal length and signal quality)

Additionally, the system does not currently print out the duration of the note. However, during the “get_note” program, the number of time instants in a note is calculated. By simply multiply this number by a sampling frequency (provided as an input from the user, potentially in the form of a tempo), the duration of a note could easily be calculate. It would then be a simple matter to print out a duration vector along with a note name vector.

Possibilities for improving upon these limitations are expanded on in the Conclusions section.

Tradeoff between signal length and quality

The current method for calculating reassigned time-frequency representation of a signal is computationally intensive. This puts a limit on the size of signal vector for which MATLAB can calculate the reassigned spectrogram (since the program requires a spectrogram frequency

resolution of 1024 frequency bins). As such there is a tradeoff between the length of the musical signal and its sampling quality. A longer signal can be recorded if it was recorded with a lower sampling frequency and vice versa (increases in quality and length both increase the length of the MATLAB vector). Most recording technology works at such a high sampling rate that the program can only analyze a few seconds of sound at the most. In order to analyze significant pieces of recognizable songs using this program, they have to be synthetically generated in order to control sampling quality.

Conclusions
The module summarizes the Conclusions from Project Treble Maker.

Conclusions

After completing the project, we concluded the following:

- Frequency reassignment is necessary for music transcription
- There is significant room to improve upon this system in order to create a comprehensive music transcription technology.

Ultimately, it's clear that this music transcription technology has great potential for expansion and could eventually become a useful tool for music lovers everywhere.

The necessity of Frequency Reassignment

After attempting music transcription with both reassigned time-frequency representations of signals and standard ones, it is clear that frequency reassignment is necessary to achieve a high level of accuracy in note identification. The "smearing" of the frequency domain that occurs with a standard representation is simply too great to provide an accurate indication of what note has been played. A "smeared" spectrum might include significant power at frequencies ranging of several distinct notes. For example, if an "E" was played, the power might be distributed across the "Eb" below that "E" and the "E" itself in a manner that prevents the "E" from being correctly identified. As such, the high frequency resolution provided by reassignment is necessary for transcription accuracy.

Expanding on the Foundation

If this project were expanded, the next steps would include expanding the range of inputs that the transcription system can handle. Obviously, music is not limited to single tonal inputs. The system could be expanded to include

chords and non tonal sounds by analyzing reassigned spectrograms of those signals and creating a template for comparison for unknown musical signals. Essentially, this would mean expanding upon the library of note “envelopes” employed by the “identify_note” MATLAB function discussed earlier to include entries for these other signals. This process would also involve modifying the algorithm for distinguishing between different notes to account for these expanded inputs.

Additionally, provision could be made for recorded signals by either improving the computational resources available (improved processor, additional MATLAB memory) or by finding a faster algorithm to compute reassigned time-frequency representations of signals.